MySql, CGI, PHP, XML, RE, CSS

1. MySQL

- A popular, fast, easy-to-use RDBMS
- Developed, marketed, and supported by MySQL AB, a Swedish company
- Released under an open-source license
- Uses a standard form of the well-known SQL language
  - Syntax may vary
  - All of our following examples work in mysql
- Works on many OS's and with many languages: PHP, PERL, C, C++, JAVA etc
- Very friendly to PHP, the most appreciated language for web development
- Supports large databases, up to 50 million rows or more in a table. Default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB)

Varies issues

- Create/delete databases
  - require root access
- Index
- MySql data types
  - INT, FLOAT, DOUBLE, DATE, DATETIME, YEAR, CHAR, VARCHAR, TEXT …

- **MySql regular expressions**
  - SELECT name FROM authors WHERE name REGEXP '^st';
  - Find all names starting with st

- Connect to PHP

2. CGI

- Static web page: same info in response to all request
- Dynamic web page: interactive, content can change in response to different contexts or conditions

- Common Gateway Interface (CGI)
  - Standard interface (protocol) through which users interact with applications on Web servers
    - Defines how information about the server and the request is passed to the command in the form of arguments and environment variables, how the command can pass back extra information about the output in the form of headers …
  - CGI script can be written in many languages: C, C++, Shell, Pascal, LISP, …
    - Perl is good at text manipulation, and the output of CGI is normally text
    - PHP and Python also popular
    - cgi-bin

3. PHP Intro

- Stands for PHP Hypertext Preprocessor
- HTML-embedded scripting language
- Linux, Windows, Mac
- Syntax borrowed from C, Java and Perl
  - a couple of unique PHP-specific features thrown in
- Goal is to allow web developers to write dynamically generated pages quickly
- As in Perl CGI, use HTML forms
- working with databases, e.g., MySql
- PHP 5 for dummies; PHP in a nutshell; PHP cookbook

Run php

- cli and cgi
- php –v;
- php hello.php
- http://cs.txstate.edu/~jg66/cgi-bin/hello.php
  - /home/Faculty/jg66/public_html/cgi-bin/hello.php

```
<?php
      echo “hello”;
?>
```

- or (no …?)

```
<?
      echo “hello”;
?>
```

php in html

```
<html>
<head><title>hello</title></head>
<body>
<?php
     echo “hello”;
?>
</body>
</html>
```

php comments

```
<html>
<head><title>hello</title></head>
<body>
<?php
      //comment1;
      # comment2;
      /* this is a
      multiple line comment
      */
      echo “hello”;
?>
</body>
</html>
```

variables

```
<html>
<head><title>hello</title></head>
<body>
<?php
      $number = 5;
      $message = “hello”;
      if ($number > 6)
        echo $message;
?>
</body>
</html>
```

double and single quoted strings

- similar to Perl. no qq though

```
<html>
<head><title>hello</title></head>
<body>
<?php
        $number = 5;
        $message = “hello”;
        if ($number > 6)
          echo “This is my statement: $message”;
?>
</body>
</html>
```

syntax similar to c++, java, perl

- arithmetic operators
- comparison operators
- if ..else
- switch
- looping
  - while, for, foreach …
- array
- associated array

sort arrays

- sort($arrayname): by value; assign new keys
- asort($arrayname): by value; keeps the same key
- rsort($arrayname): by value in reverse order, assign new keys
- arsort($arrayname): by value in reverse order, keeps key
- ksort($arrayname): by key
- krsort($arrayname): by key in reverse order
- usort($arrayname, functionname): by a function

more on arrays

- exchanging keys and values

```
$arrayFlipped = array_flip($originalArray);
```

- finding array size

```
$n = count($arrayname);
$n = sizeof()$arrayname;
```

- iterating

```
foreach ( $arrayname as $keyname => $valuename )
{
  block of statements;
}
```

- spliting, merging
- multi-dimensional

php function

```
<?php
   function myGreeting($firstName){
       echo "Hello there, ". $firstName;
   }

   myGreeting("Jack");
   myGreeting("Charles");
?>
```

- can also return values

forms

```
<html><body>
<form action="form.php" method="post">
<select name="item">
<option>Easy Questions</option>
<option>Hard Questions</option>
</select>
Quantity: <input name="quantity" type="text" />
<input name="submit" type="submit" />
</form>
<?php
if($_POST['submit']) {
   $quantity = $_POST['quantity'];
   $item = $_POST['item'];
   echo "You ordered ". $quantity . " " . $item . ".<br>";
   echo "Thank you for the ordering!";
}
?>
</body></html>
```

retrieve the submitted info from associated array $_POST

files

```
$fh = fopen(“file1.txt”,”r”);
while(!feof($fh))
{
   $line = fgets($fh);
   echo “$line”;
}
```

- feof($fh) returns true when the end of file is reached

```
$fh = fopen(“file1.txt”,”w”);
fwrite($fh, “something”);


fclose($fh);
```

more on reading files

```
$fh = fopen(“file2.txt”,”r”);
while(!feof($fh))
{
   $content[] = fgets($fh);
}
fclose($fh);
```

- array $content, with each line of file as element

```
$content = file(“file2.txt”);
```

- short cut

```
$stringContent = file_get_contents(“file2.txt”,1);
```

- reading file into a string

pattern matching

```
$string = ‘one tree, two trees, three trees’;

if (ereg(‘tree’, $string)) { … }

ereg_replace(‘tree’, ‘frog’, $string);
```

- find tree in $string and replace it with frog
- returns ‘one frog, two frogs, three frogs’

working with mysql: testmysql.php

```
<html><head><title>Testing MySql</title></head><body>

<?php

$host ='mysql.cs.txstate.edu';
$user = 'jg66';
$password = ‘blabla';
$dbname = 'jg66';

$connection = mysql_connect($host,$user,$password) or die ("Couldn't connect to server");
$db = mysql_select_db($dbname,$connection) or die ("Couldn't select database");
$query = "SELECT * FROM Sailors";
$result = mysql_query($query) or die("Query failed: ".mysql_error());
  # $result contains a pointer to a temporary table of results with rows and columns
while ($row = mysql_fetch_array($result)) # retrieve one row and store in an array $row
{
  echo $row['sname'];
  echo "<br>";
}
mysql_close($connection);
?>

</body></html>
```

modified testmysql.php

```
<html><head><title>Testing MySql</title></head><body>

<?php

include('info.inc'); # contains connect variables.
$connection = mysql_connect($host,$user,$password) or die ("Couldn't connect to server");
$db = mysql_select_db($dbname,$connection) or die ("Couldn't select database");
$query = "SELECT * FROM Sailors";
$result = mysql_query($query) or die("Query failed: ".mysql_error());
    # $result contains a pointer to a temporary table of results with rows and columns
while ($row = mysql_fetch_array($result)) # retrieve one row and store in an array $row
{
  echo $row['sname'];
  echo "<br>";
}
mysql_close($connection);
?>

</body></html>
```

info.inc

```
<?php
$host ='mysql.cs.txstate.edu';
$user = 'jg66';
$password = 'blabla';
$dbname = 'jg66';
?>
```

phpinfo.php

```
<?php
// Show all information, defaults to INFO_ALL
phpinfo();

// Show just the module information.
// phpinfo(8) yields identical results.
phpinfo(INFO_MODULES);
?>
```

- /home/Faculty/jg66/public_html/cgi-bin/phpinfo.php
- http://www.cs.txstate.edu/~jg66/cgi-bin/phpinfo.php

upload file in php

- http://www.tizag.com/phpT/fileupload.php

- Create uploads sub-directory
- Grant write permission

- Example, twobox.php

4. XML

- eXtensible Markup Language
- Markup language like HTML, but different
- Designed to transport and store data, not to display data
- Tags are not predefined. You define your own tags
- Self descriptive
- In plain text, compatible, software and hardware independent
  - Exchanging data in XML greatly reduces complexity
  - drawback?
- XML style

5. Regular expression

- Regular expressions are sets of symbols and syntactic elements used to match patterns of text
  - provide a concise and flexible means for identifying strings of text of interest
  - template
  - popular search tool grep
  - expr, AWK, Emacs, vi, lex
  - POSIX
  - Perl derivative
    - PHP, Java, JavaScript, Pythin, Ruby, .Net
  - testing; many online tools http://www.nvcc.edu/home/drodgers/CEU/Resources/test_regexp.asp

Basic Syntax

Char	Usage	Example
.	Matches any single character	.at = cat, bat, rat, 1at…
*****	Matches zero or more occurrences of the single preceding character	.*at = everything that ends with at 0*123 = 123, 0123, 00123…
[…] **[^…]**	Matches any single character of the ones contained Matches any single character except for the ones contained	[cbr]at = cat, bat, rat. [^bc]at = rat, sat…, *but not* bat, cat. <[^>]*> = <…anything…>
^	Beginning of line	^a = line starts with a
$	End of line	^$ = blank line (starts with the end of line)
****	Escapes following special character: . \ / & [ ] * + -> \. \\ \/ \& \[\] * \+	[cbr]at\. = matches cat., bat. and rat. only
…	…	

Character Classes

[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z, or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z] (subtraction)

Predefined Character Classes

.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Quantifiers

Contains number of times a preceding character appear

Greedy	Meaning
X?	X, 0 or 1
X*	X, 0 or more
X+	X, 1 or more
X{n}	X, exactly n times
X{n,}	X, at least n times
X{n,m}	X, at least n but not more than m times

Groups

- With parentheses, we can create groups to apply quantifiers to several characters: “(abc)+”
- Also useful in parsing
- Groups are numbered by counting their opening parentheses from left to right
- Example: groups in “((A)(B(C)))”
  1. ((A)(B(C)))
  2. (A)
  3. (B(C))
  4. (C)
- $1, $2, $3, $4

Boundary matchers

^	The beginning of a line
$	The end of a line
\b	A word boundary
\B	A non-word boundary
\A	The beginning of the input
\G	The end of the previous match
\Z	The end of the input but for the final terminator, if any
\z	The end of the input

Current REGEX is: \bdog\b
Current INPUT is: The doggie plays in the yard.
No match found.

6. CSS

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Separate content of HTML documents from presentation layout
- **External Style Sheets** can save you a lot of work
  - Control multiple webpages all at once
- External Style Sheets are stored in **CSS files**
  - styles.css
- Multiple style definitions will **cascade** into one with order:
  1. Browser default
  2. External style sheet
  3. Internal style sheet (inside the <head> tag)
  4. Inline style (inside an HTML element)

Insert external style sheet

```
<head>
 <link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

- Be aware of IE compatibility issues

```
<head>
 <title>Byron Gao's Homepage</title>
<link rel="stylesheet" type="text/css" href="styles.css" />
<!--[if IE 7]><link rel="stylesheet" type="text/css"
    href="IE7styles.css" /><![endif]-->
</head>
```

example mystyle.css

```
hr {height: 1px;}
p {
  color: red;
  margin-left: 20px;
}
body {background: white;}
```

Internal style sheet

- Used when a single document has a unique style
- Defined in the head section by using the <style> tag

```
<head>
<style type="text/css">
hr {height: 1px;}
p {
  color: red;
  margin-left: 20px;
}
body {background: white;}
</style>
</head>
```

Inline styles

- Inline style loses the advantages of style sheets by mixing content with presentation
- Used when a style is to be applied to a single occurrence of an element
- Use the style attribute in the relevant tag

<p style=“color: red; margin-left: 20px">
This is a paragraph
</p>

Syntax

- selector {property: value}
  - body {color: black}
  - use ; if more than one property

```
p {
        color: red;
        margin-left: 20px;
}
```

  - Comment. /* this is for … */

- Grouping: separate selector with comma

```
h1,h2,h3,h4,h5,h6
{ color: green }
```

Class selector

- Class selector allows you define different styles for the same type of HTML element. E.g., two types of paragraphs

  p.right {text-align: right}

  p.center {text-align: center}

- In HTML document:

  This paragraph will be right-aligned.

  This paragraph will be center-aligned.

More on class selector

- Can omit tag name in the selector to define a style to be used by all HTML elements of the class. E.g., to have all HTML elements with class="center“ center-aligned:

  .center {text-align: center}

- In HTML document:

```
<h1 class="center">
This heading will be center-aligned
</h1>

<p class="center">
This paragraph will also be center-aligned.
</p>
```

id selector

- Can also define styles for HTML elements with id selector

```
p#para1 {
  text-align: right;
  color: red;
}
```

- In HTML document:

```
<p id="para1">
This paragraph will be right-aligned and in red.
</p>
```

- Can also omit tag name in the selector to match any HTML elements with the id:

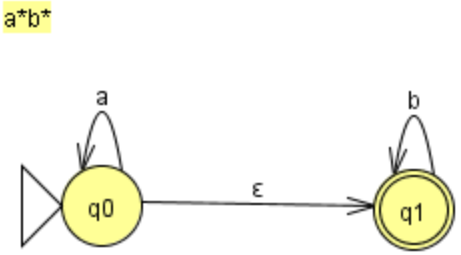
```
#green {color: green}
```

- #leftcolumn { … } In HTML: …

examples

- Homepage

7. JFLAP



- What's JFLAP? http://www.jflap.org/whatis.html
- Download, tutorial: http://www.jflap.org/
- Can also use applet: http://www.cs.duke.edu/csed/jflap/jflaptmp/applet/demo.html
- Preferences: set the empty string character to episilon
- FSM, TM, Mealy, all fine. Just PDA has different definition from ours:
  - with Z, a stack marker (we don't have it)
  - Either finite state or empty stack acceptance (we use both)
  - To make our PDAs run in JFLAP: choose acceptance option properly. Sometimes may need to remove Z.
- To run a machine: step (step with closure for ND), fast run, **multiple run**
- Grammar:
  - Test for grammar type
  - Brute force parse, multiple brute force parse
  - Convert

(to be added)